#Write a program to implement Linear Regression (LR) algorithm in python

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

import matplotlib.pyplot as plt

# Load the dataset

dataset = pd.read\_csv('Salary\_Data.csv')

# Split the dataset into independent variables (X) and dependent variable (y)

X = dataset.iloc[:, :-1].values

y = dataset.iloc[:, -1].values

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Create an instance of the Linear Regression model

model = LinearRegression()

# Fit the model to the training data

model.fit(X\_train, y\_train)

# Predict the salaries for the test data

y\_pred = model.predict(X\_test)

#model good or not

mse = mean\_squared\_error(y\_test, y\_pred)

print("Mean Squared Error:", mse)

#Visualising the Training set results Here scatter plot is used to visualize the results.

plt.scatter(X\_train, y\_train, color = 'red')

plt.plot(X\_train, model.predict(X\_train), color = 'blue')

plt.title('Salary vs Experience (Training set)')

plt.xlabel('Years of Experience')

plt.ylabel('Salary')

plt.show()

LogisticRegression

#Import Libraries

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import mean\_squared\_error

import matplotlib.pyplot as plt

#Read data set

dataset = pd.read\_csv("D:/GEO/BE COURSES/2022 dec/LAB/DATASET/breastcancer.csv")

# Separate X & y

X = dataset.iloc[:, :-1].values

y = dataset.iloc[:, -1].values

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

#Feature scaling is the process of converting the data into a given range. In this case, the standard scalar technique is used.

model = LogisticRegression(random\_state = 0)

model.fit(X\_train, y\_train)

# Predict the salaries for the test data

y\_pred = model.predict(X\_test)

#confusion matrix of thee model

from sklearn.metrics import confusion\_matrix, accuracy\_score

cm = confusion\_matrix(y\_test, y\_pred)

print(cm)

print('Accuracy Score:confusion matrix')

# Calculate the accuracy of the model

accuracy = accuracy\_score(y\_test, y\_pred)

print("Accuracy:", accuracy)

# -\*- coding: utf-8 -\*-

"""

Created on Tue Jun 20 10:36:42 2023

**Linear and poly:no need of splitting data**

@author: GEOMOL GEORGE

"""

#Import Libraries

import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import LogisticRegression

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

import matplotlib.pyplot as plt

#Read data set

dataset = pd.read\_csv('Position\_Salaries.csv')

# Separate X & y

X = dataset.iloc[:, 1:-1].values

y = dataset.iloc[:, -1].values

#split data

#X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Create an instance of the Linear Regression model

model = LinearRegression()

# Fit the model to the training data

model.fit(X,y)

#Visualising the Linear Regression results

plt.scatter(X, y, color = 'red')

plt.plot(X, model.predict(X), color = 'blue')

plt.title(' (Linear Regression)')

plt.xlabel('Position Level')

plt.ylabel('Salary')

plt.show()

#model polynomial

from sklearn.preprocessing import PolynomialFeatures

model= PolynomialFeatures(degree = 4)

poly=model.fit\_transform(X)

model3 = LinearRegression()

model3.fit(poly, y)

#Visualising the Polynomial Regression results

plt.scatter(X, y, color = 'red')

plt.plot(X, model3.predict(model.fit\_transform(X)), color = 'blue')

plt.title('Truth or Bluff (Polynomial Regression)')

plt.xlabel('Position level')

plt.ylabel('Salary')

plt.show()